

Termination Resilience Analysis

**MPRI 2-6: Abstract Interpretation,
Application to Verification and Static Analysis**

Caterina Urban

November 18th, 2024

Year 2024-2025

Language Syntax

Example

stat ::= $\ell X \leftarrow \text{input}$

$\ell X \leftarrow \text{exp}$

if $\ell \text{exp} \bowtie 0$ **then** stat

while $\ell \text{exp} \bowtie 0$ **do** stat **done**

stat ; stat

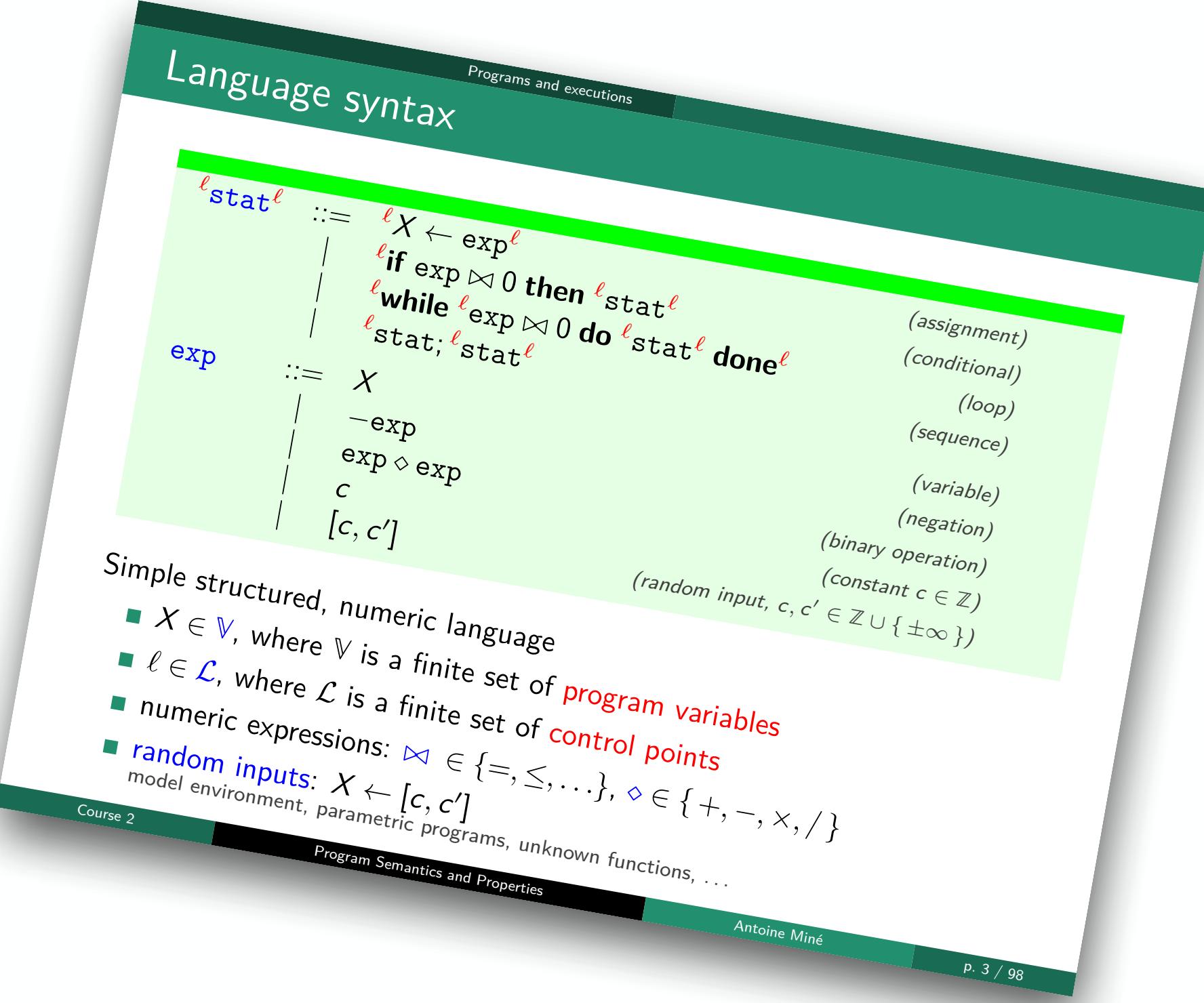
1 $x \leftarrow [-\infty, +\infty]$

2 $y \leftarrow \text{input}$

while **3** ($x > 0$) **do**

4 $x \leftarrow x - y$

od**5**



Transition Systems

Example

1 $x \leftarrow [-\infty, +\infty]$

2 $y \leftarrow \text{input}$

while **3** ($x > 0$) **do**

4 $x \leftarrow x - y$

od⁵

$$\Sigma \stackrel{\text{def}}{=} \{1, 2, 3, 4, 5\} \times \mathcal{E}$$

$$\begin{aligned} \tau \stackrel{\text{def}}{=} & \{(1, \rho) \rightarrow (2, \rho[x \mapsto v]) \mid \rho \in \mathcal{E}, v \in \mathbb{Z}\} \\ & \cup \{(2, \rho) \rightarrow (3, \rho[y \mapsto v]) \mid \rho \in \mathcal{E}, v \in \mathbb{Z}\} \\ & \cup \{(3, \rho) \rightarrow (4, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[x] \rho: v > 0\} \\ & \cup \{(4, \rho) \rightarrow (3, \rho[x \mapsto v]) \mid \rho \in \mathcal{E}, v \in E[x - y] \rho\} \\ & \cup \{(3, \rho) \rightarrow (5, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[x] \rho: v \leq 0\} \end{aligned}$$

Program Semantics and Properties

Antoine Miné

P. 8 / 98

From programs to transition relations

Programs and executions

Transitions: $\tau[\ell^{\text{stat}}]^{\ell'} \subseteq \Sigma \times \Sigma$

$\tau[\ell^1 X \leftarrow e^{\ell^2}] \stackrel{\text{def}}{=} \{(\ell^1, \rho) \rightarrow (\ell^2, \rho[X \mapsto v]) \mid \rho \in \mathcal{E}, v \in E[e] \rho\}$

$\tau[\ell^1 \text{if } e \triangleq 0 \text{ then } \ell^2 s^{\ell^3}] \stackrel{\text{def}}{=} \{(\ell^1, \rho) \rightarrow (\ell^2, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[e] \rho: v \triangleq 0\} \cup \{(\ell^1, \rho) \rightarrow (\ell^3, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[e] \rho: v \not\triangleq 0\} \cup \tau[\ell^2 s^{\ell^3}]$

$\tau[\ell^1 \text{while } \ell^2 e \triangleq 0 \text{ do } \ell^3 s^{\ell^4} \text{ done}^{\ell^5}] \stackrel{\text{def}}{=} \{(\ell^1, \rho) \rightarrow (\ell^2, \rho) \mid \rho \in \mathcal{E}\} \cup \{(\ell^2, \rho) \rightarrow (\ell^3, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[e] \rho: v \triangleq 0\} \cup \{(\ell^4, \rho) \rightarrow (\ell^2, \rho) \mid \rho \in \mathcal{E}\} \cup \{(\ell^2, \rho) \rightarrow (\ell^5, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[e] \rho: v \not\triangleq 0\} \cup \tau[\ell^3 s^{\ell^4}] \cup \tau[\ell^4 s^{\ell^5}]$

$\tau[\ell^1 s_1; \ell^2 s_2]^{\ell^3} \stackrel{\text{def}}{=} \tau[\ell^1 s_1 \ell^2] \cup \tau[\ell^2 s_2 \ell^3]$

(expression semantics $E[e]$ on next slide)

Termination Resilience

Potential Termination

Definition

A program with trace semantics
 $\mathcal{M} \in \mathcal{P}(\Sigma^\infty)$ **may terminate**
if and only if $\mathcal{M} \cap \Sigma^* \neq \emptyset$

Definite Termination

Definition

A program with trace semantics
 $\mathcal{M} \in \mathcal{P}(\Sigma^\infty)$ **may terminate**
if and only if $\mathcal{M} \cap \Sigma^* \neq \emptyset$

$$\text{PotentialTermination} \stackrel{\text{def}}{=} \{T \in \mathcal{P}(\Sigma^\infty) \mid \exists t \in T: t \in \Sigma^*\}$$

$$\text{DefiniteTermination} \stackrel{\text{def}}{=} \{T \in \mathcal{P}(\Sigma^\infty) \mid \forall t \in T: t \in \Sigma^*\}$$

$$\text{TerminationResilience} \stackrel{\text{def}}{=} \{T \in \mathcal{P}(\Sigma^\infty) \mid \forall i \in \text{input values read in a traces } t \in T : \exists t \in T : t^i = i \wedge t \in \Sigma^*\}$$

input values read in a traces t
input values read in a set of traces T

Termination Resilience

Example

```

1 x ← [-∞, +∞]
2 y ← input
while 3(x > 0) do
    4 x ← x - y
od5

```

$\mathcal{M}_\infty \in \text{TerminationResilience}$

$y > 0$: the program always terminates

$y \leq 0$: the program terminates when $x \leq 0$

Maximal traces

Maximal traces:

$$\mathcal{M}_\infty \in \mathcal{P}(\Sigma^\infty)$$

- sequences of states linked by the transition relation τ
- start in any state ($\mathcal{I} = \Sigma$, technical requirement for the fixpoint characterization)
- either finite and stop in a blocking state ($\mathcal{F} = \mathcal{B}$)
- or infinite

$$\mathcal{M}_\infty \stackrel{\text{def}}{=} \left\{ \sigma_0, \dots, \sigma_n \in \Sigma^* \mid \sigma_n \in \mathcal{B}, \forall i < n : \sigma_i \rightarrow \sigma_{i+1} \right\} \cup \\ \left\{ \sigma_0, \dots, \sigma_n, \dots \in \Sigma^\omega \mid \forall i < \omega : \sigma_i \rightarrow \sigma_{i+1} \right\}$$

(can be anchored at \mathcal{I} and \mathcal{F} as: $\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \cap ((\Sigma^* \cdot \mathcal{F}) \cup \Sigma^\omega)$)

Course 2

Program Semantics and Properties

Maximal trace semantics

Antoine Miné

p. 72 / 98

Least fixpoint formulation of maximal traces

Idea: To get a least fixpoint formulation for whole \mathcal{M}_∞ , we merge finite and infinite maximal trace least fixpoint forms

Fixpoint fusion:

$\mathcal{M}_\infty \cap \Sigma^*$ is best defined on $(\mathcal{P}(\Sigma^*), \subseteq, \cup, \cap, \emptyset, \Sigma^*)$.
 $\mathcal{M}_\infty \cap \Sigma^\omega$ is best defined on $(\mathcal{P}(\Sigma^\omega), \supseteq, \cap, \cup, \Sigma^\omega, \emptyset)$, the dual lattice.
(we transform the greatest fixpoint into a least fixpoint!)

We mix them into a new complete lattice $(\mathcal{P}(\Sigma^\infty), \sqsubseteq, \sqcup, \sqcap, \perp, \top)$:

- $A \sqsubseteq B \stackrel{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \wedge (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$
- $A \sqcup B \stackrel{\text{def}}{=} ((A \cap \Sigma^*) \cup (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cap (B \cap \Sigma^\omega))$
- $A \sqcap B \stackrel{\text{def}}{=} ((A \cap \Sigma^*) \cap (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cup (B \cap \Sigma^\omega))$
- $\perp \stackrel{\text{def}}{=} \Sigma^\omega$
- $\top \stackrel{\text{def}}{=} \Sigma^*$

In this lattice, $\mathcal{M}_\infty = \text{lfp } F_s$ where $F_s(T) \stackrel{\text{def}}{=} \mathcal{B} \cup \tau \cap T$

(proof on next slides)

Course 2

Program Semantics and Properties

Antoine Miné

p. 76 / 98

Termination Resilience Semantics

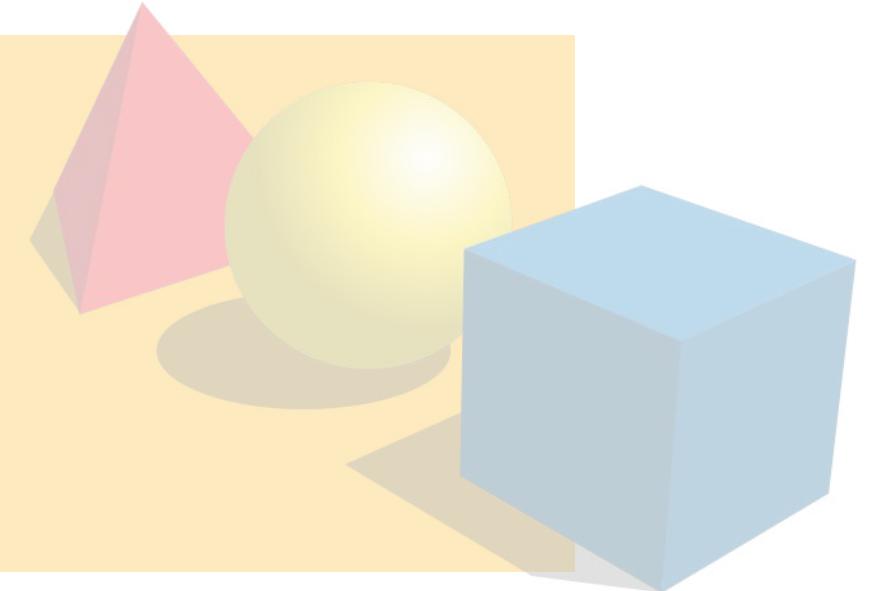
practical tools

targeting specific programs



algorithmic approaches

to decide program properties



mathematical models

of the program behavior



Termination Resilience Semantics

Potential Termination Semantics

$$\mathcal{R}_m = \text{lfp}^{\leq} F_m$$

$$F_m(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \\ v & \sigma \in \text{pre}_{\tau}(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$v \stackrel{\text{def}}{=} \inf\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\}$$

Definite Termination Semantics

$$\mathcal{R}_M = \text{lfp}^{\leq} F_M$$

$$F_M(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \\ \bar{v} & \sigma \in \tilde{\text{pre}}_{\tau}(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\bar{v} \stackrel{\text{def}}{=} \sup\{f(\sigma') + 1 \mid (\sigma, \sigma') \in \tau\}$$

$$\mathcal{R}_{TR} = \text{lfp}^{\leq} F_{TR}$$

$$F_{TR}(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \quad \text{input state transitions} \\ \bar{v} & \sigma \in \tilde{\text{pre}}_{\tau^i}(\text{dom}(f)) \\ v & \sigma \in \text{pre}_{\tau^r}(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \quad \text{regular state transitions} \end{cases}$$

Denotational Termination Resilience Semantics

For each program instruction stat, we define a transformer $\mathcal{R}_{TR}[[\text{stat}]] : (\mathcal{E} \rightarrow \mathbb{O}) \rightarrow (\mathcal{E} \rightarrow \mathbb{O})$:

- $\mathcal{R}_{TR}[[^\ell X \leftarrow \text{input}]]$
- $\mathcal{R}_{TR}[[^\ell X \leftarrow e]]$
- $\mathcal{R}_{TR}[[\text{if } ^\ell e \bowtie 0 \text{ then } s]]$
- $\mathcal{R}_{TR}[[\text{while } ^\ell e \bowtie 0 \text{ do } s \text{ done}]]$
- $\mathcal{R}_{TR}[[s_1; s_2]]$

Denotational Termination Resilience Semantics

$\mathcal{R}_{TR}[\![^\ell X \leftarrow \text{input}]\!]$

$$\mathcal{R}_{TR}[\![^\ell X \leftarrow \text{input}]\!]f \stackrel{\text{def}}{=} \lambda\rho. \begin{cases} \sup\{f(\rho[X \mapsto v]) + 1 \mid v \in E[\![e]\!]\rho\} & \forall v \in \mathbb{Z}: \rho[X \mapsto v] \in \text{dom}(f) \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\mathcal{R}_{TR} = \text{lfp}^{\leq} F_{TR}$$

$$F_{TR}(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \\ \bar{v} & \sigma \in \tilde{\text{pre}}_{\tau^i}(\text{dom}(f)) \\ v & \sigma \in \text{pre}_{\tau^r}(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Denotational Termination Resilience Semantics

$\mathcal{R}_{TR}[\![\ell X \leftarrow e]\!]$

the value of e (and thus of X) depends from the input values read by the program

$$\mathcal{R}_{TR}[\![\ell X \leftarrow e]\!]f \stackrel{\text{def}}{=} \lambda\rho . \begin{cases} \sup\{f(\rho[X \mapsto v]) + 1 \mid v \in E[\![e]\!]\rho\} & \mathcal{T}[\![e]\!]\mathcal{T}(\ell) \wedge \text{all} \\ \inf\{f(\rho[X \mapsto v]) + 1 \mid v \in E[\![e]\!]\rho\} & \neg\mathcal{T}[\![e]\!]\mathcal{T}(\ell) \wedge \text{any} \\ \text{undefined} & \text{otherwise} \end{cases}$$

all $\stackrel{\text{def}}{=} \exists \bar{v} \in E[\![e]\!]\rho \wedge \forall v \in E[\![e]\!]\rho : \rho[X \mapsto v] \in \text{dom}(f)$

any $\stackrel{\text{def}}{=} \exists v \in E[\![e]\!]\rho : \rho[X \mapsto v] \in \text{dom}(f)$

$$\mathcal{R}_{TR} = \text{lfp}^{\leq} F_{TR}$$

$$F_{TR}(f)\sigma \stackrel{\text{def}}{=} \begin{cases} 0 & \sigma \in \mathcal{B} \\ \bar{v} & \sigma \in \tilde{\text{pre}}_{\tau^i}(\text{dom}(f)) \\ v & \sigma \in \text{pre}_{\tau^r}(\text{dom}(f)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Simple Taint Semantics

$$\mathcal{T}[\![\text{stat}]\!]: \mathcal{P}(\mathbb{X}) \rightarrow \mathcal{P}(\mathbb{X})$$

$$\mathcal{T}[\![\ell X \leftarrow \text{input}]\!]T \stackrel{\text{def}}{=} T \cup \{X\}$$

$$\mathcal{T}[\![\ell X \leftarrow e]\!]T \stackrel{\text{def}}{=} \begin{cases} T \cup \{X\} & \mathcal{T}[\![e]\!]T \\ T \setminus \{X\} & \text{otherwise} \end{cases}$$

$$\mathcal{T}[\![\text{if } \ell e \bowtie 0 \text{ then } s]\!]T \stackrel{\text{def}}{=} \begin{cases} \mathcal{T}[\![s]\!]T \cup \text{LHS}(s) \cup T & \mathcal{T}[\![e]\!]T \\ \mathcal{T}[\![s]\!]T \cup T & \text{otherwise} \end{cases}$$

assigned variables in s

$$\mathcal{T}[\![\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]\!]T \stackrel{\text{def}}{=} \text{lfp}_T^{\subseteq} \left(\lambda X. \begin{cases} \mathcal{T}[\![s]\!]X \cup \text{LHS}(s) \cup X & \mathcal{T}[\![e]\!]X \\ \mathcal{T}[\![s]\!]X \cup X & \text{otherwise} \end{cases} \right)$$

$$\mathcal{T}[\![s_1; s_2]\!]T \stackrel{\text{def}}{=} \mathcal{T}[\![s_2]\!](\mathcal{T}[\![s_1]\!]T)$$

$$\mathcal{T}[\![e]\!]T \stackrel{\text{iff}}{\Leftrightarrow} \exists \rho \in \mathcal{E}: \exists v \in E[\![e]\!]\rho: \forall \rho' \in \mathcal{E}: \rho' \neq_T \rho \Rightarrow v \notin E[\![e]\!]\rho'$$

$$\rho' \neq_T \rho \equiv \forall x \notin T: \rho'(x) = \rho(x) \wedge \exists x \in T: \rho'(x) \neq \rho(x)$$

Language syntax

ℓstat^ℓ	$::= \ell X \leftarrow \text{exp}^\ell$	(assignment)
	$\mid \ell \text{if } \text{exp} \bowtie 0 \text{ then } \ell \text{stat}^\ell$	(conditional)
	$\mid \ell \text{while } \ell \text{exp} \bowtie 0 \text{ do } \ell \text{stat}^\ell$	(loop)
	$\mid \ell \text{stat}; \ell \text{stat}^\ell$	(sequence)
exp	$::= X$	(variable)
	$\mid -\text{exp}$	(negation)
	$\mid \text{exp} \diamond \text{exp}$	(binary operation)
	$\mid c$	(constant $c \in \mathbb{Z}$)
	$\mid [c, c']$	(random input, $c, c' \in \mathbb{Z} \cup \{\pm\infty\}$)

Simple structured, numeric language

- $X \in \mathbb{V}$, where \mathbb{V} is a finite set of program variables
- $\ell \in \mathcal{L}$, where \mathcal{L} is a finite set of control points
- numeric expressions: $\bowtie \in \{=, \leq, \dots\}$, $\diamond \in \{+, -, \times, /\}$
- random inputs: $X \leftarrow [c, c']$ model environment, parametric programs, unknown functions, ...

Course 2 Program Semantics and Properties Antoine Miné p. 3 / 98

Denotational Termination Resilience Semantics

$\mathcal{R}_{TR}[\![\text{if } \ell e \bowtie 0 \text{ then } s]\!]$

$$\mathcal{R}_{TR}[\![\text{if } \ell e \bowtie 0 \text{ then } s]\!]f \stackrel{\text{def}}{=} \lambda\rho . \begin{cases} \textcircled{1} & \exists v_1, v_2 \in E[\![e]\!] \rho : v_1 \bowtie 0 \wedge v_2 \bowtie 0 \\ \textcircled{2} & \exists v_1, v_2 \in E[\![e]\!] \rho : v_1 \bowtie 0 \wedge v_2 \bowtie 0 \\ \mathcal{R}_{TR}[\![s]\!]f(\rho) + 1 & \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]f) \wedge \forall v \in E[\![e]\!] \rho : v \bowtie 0 \\ f(\rho) + 1 & \rho \in \text{dom}(f) \wedge \forall v \in E[\![e]\!] \rho : v \bowtie 0 \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\textcircled{1} \quad \sup\{\mathcal{R}_{TR}[\![s]\!]f(\rho) + 1, f(\rho) + 1\} \quad \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]f) \cap \text{dom}(f)$$

$$\begin{array}{ll} \textcircled{2} \quad \inf\{\mathcal{R}_{TR}[\![s]\!]f(\rho) + 1, f(\rho) + 1\} & \neg \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]f) \cap \text{dom}(f) \\ & \neg \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]f) \setminus \text{dom}(f) \\ \mathcal{R}_{TR}[\![s]\!]f(\rho) + 1 & \neg \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(f) \setminus \text{dom}(\mathcal{R}_{TR}[\![s]\!]f) \\ f(\rho) + 1 & \end{array}$$

Denotational Termination Resilience Semantics

$\mathcal{R}_M[\![\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]\!]$

$$\mathcal{R}_{TR}[\![\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]\!]f \stackrel{\text{def}}{=} \text{lfp}_{\preceq_{\emptyset}} \left(\lambda x \rho . \begin{cases} \textcircled{1} & \exists v_1, v_2 \in E[\![e]\!] \rho : v_1 \bowtie 0 \wedge v_2 \bowtie 0 \\ \textcircled{2} & \exists v_1, v_2 \in E[\![e]\!] \rho : v_1 \bowtie 0 \wedge v_2 \bowtie 0 \\ \mathcal{R}_{TR}[\![s]\!]x(\rho) + 1 & \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]x) \wedge \forall v \in E[\![e]\!] \rho : v \bowtie 0 \\ f(\rho) + 1 & \rho \in \text{dom}(f) \wedge \forall v \in E[\![e]\!] \rho : v \bowtie 0 \\ \text{undefined} & \text{otherwise} \end{cases} \right)$$

$$\textcircled{1} \quad \sup\{\mathcal{R}_{TR}[\![s]\!]x(\rho) + 1, f(\rho) + 1\} \quad \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]x) \cap \text{dom}(f)$$

$$\begin{aligned} \textcircled{2} \quad \inf\{\mathcal{R}_{TR}[\![s]\!]x(\rho) + 1, f(\rho) + 1\} & \quad \neg \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]x) \cap \text{dom}(f) \\ \mathcal{R}_{TR}[\![s]\!]x(\rho) + 1 & \quad \neg \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(\mathcal{R}_{TR}[\![s]\!]x) \setminus \text{dom}(f) \\ f(\rho) + 1 & \quad \neg \mathcal{T}[\![e]\!] \mathcal{T}(\ell) \wedge \rho \in \text{dom}(f) \setminus \text{dom}(\mathcal{R}_{TR}[\![s]\!]x) \end{aligned}$$

Denotational Termination Resilience Semantics

$\mathcal{R}_M[s_1; s_2]$

$\mathcal{R}_{TR}[s_1; s_2]f \stackrel{\text{def}}{=} \mathcal{R}_{TR}[s_1](\mathcal{R}_{TR}[s_2]f)$

Denotational Termination Resilience Semantics

Definition

The **denotational termination resilience semantics** $\mathcal{R}_{TR}[\![\text{stat}^{\ell}]\!]: \mathcal{E} \rightarrow \mathbb{O}$ of a program stat^{ℓ} is:

$$\mathcal{R}_{TR}[\![\text{stat}^{\ell}]\!] \stackrel{\text{def}}{=} \mathcal{R}_{TR}[\![\text{stat}]\!](\lambda\rho.0)$$

where $\mathcal{R}_{TR}[\![\text{stat}]\!]: (\mathcal{E} \rightarrow \mathbb{O}) \rightarrow (\mathcal{E} \rightarrow \mathbb{O})$ is the denotational termination resilience semantics of each program instruction stat

Theorem (Soundness and Completeness)

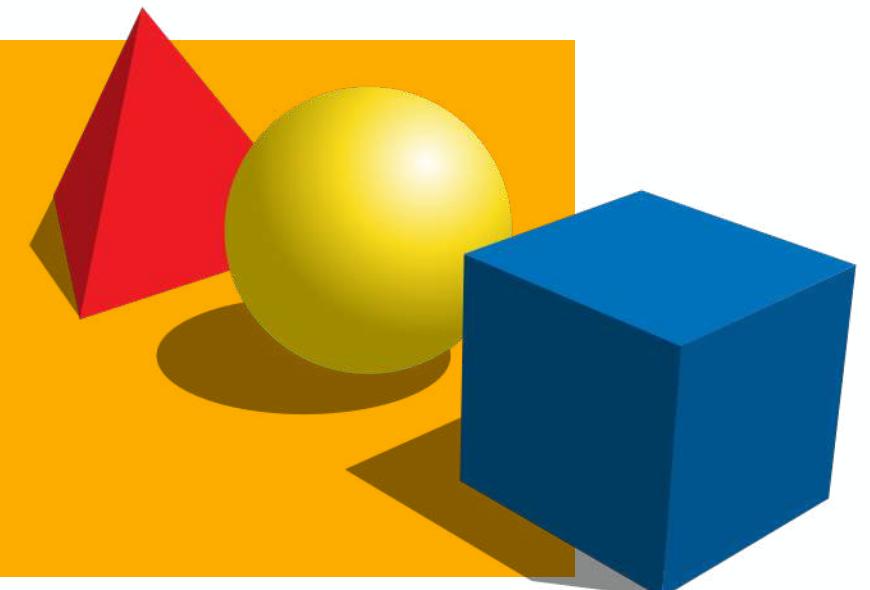
A program stat^{ℓ} **satisfies termination resilience** for traces starting from a set of initial states \mathcal{I} , i.e., $\mathcal{M}_{\infty}[\![\text{stat}^{\ell}]\!](\mathcal{I}) \in \text{TerminationResilience}$, if and only if $\mathcal{I} \subseteq \text{dom}(\mathcal{R}_{TR}[\![\text{stat}^{\ell}]\!])$

Piecewise-Defined Ranking Functions

practical tools
targeting specific programs



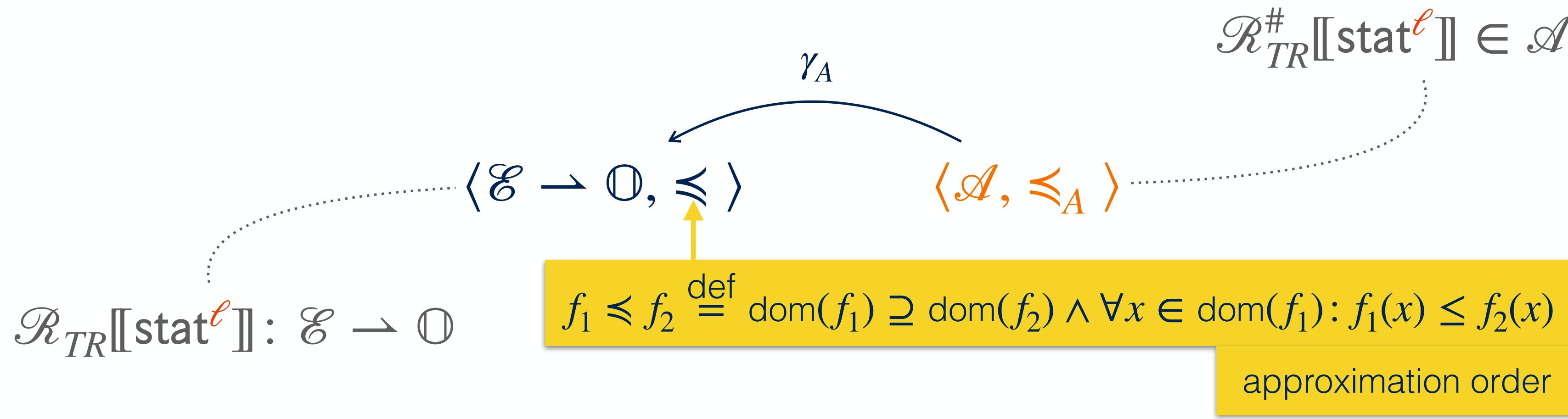
algorithmic approaches
to decide program properties



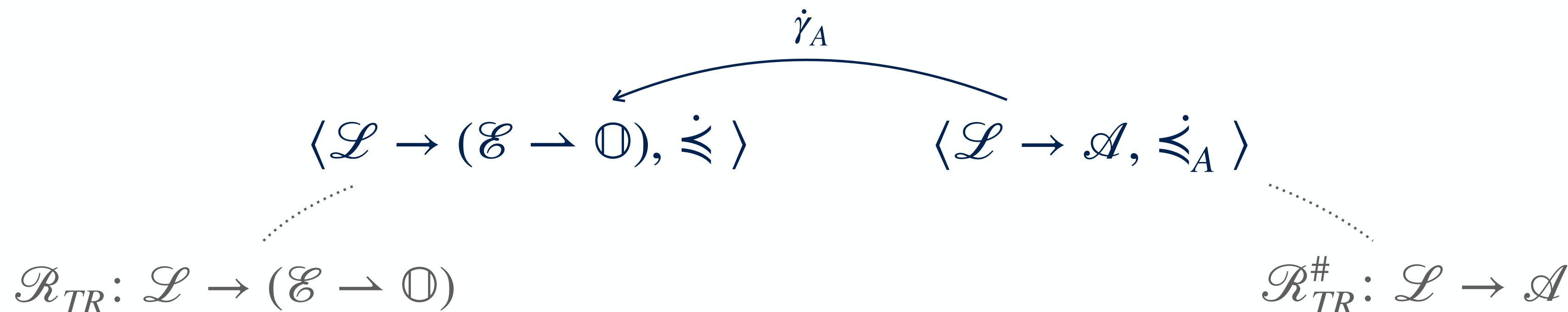
mathematical models
of the program behavior



Concretization-Based Piecewise Abstraction



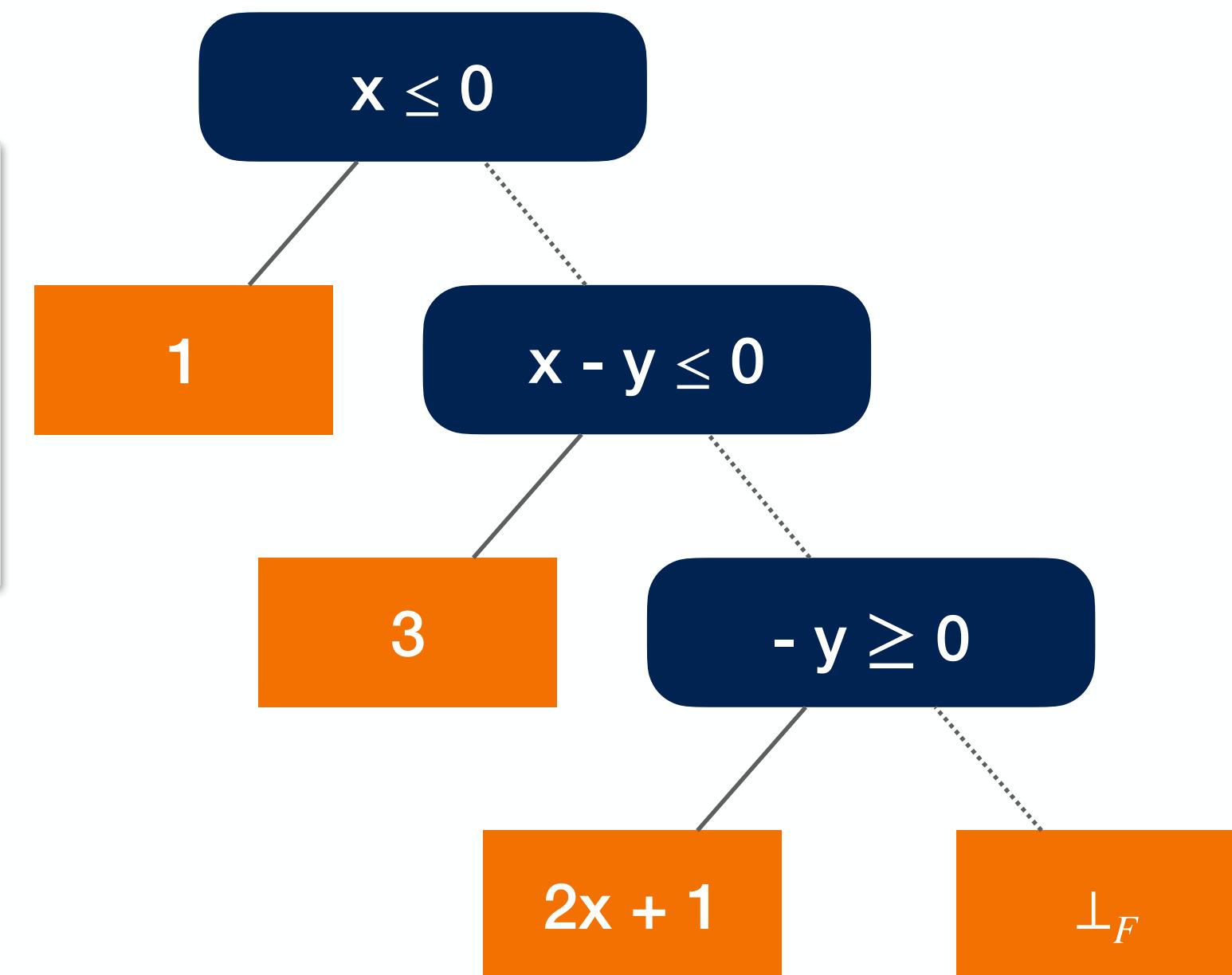
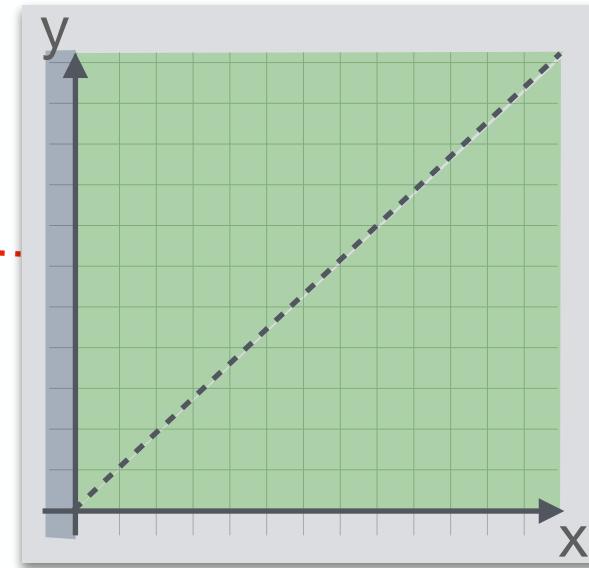
By *pointwise lifting* we obtain an abstraction $\mathcal{R}_{TR}^\#$ of \mathcal{R}_{TR} :



Piecewise-Defined Ranking Functions Abstract Domain

$$\mathcal{A} \stackrel{\text{def}}{=} \{\text{LEAF}: f \mid f \in \mathcal{F}\} \cup \{\text{NODE}\{c\}: t_1; t_2 \mid c \in \mathcal{C} \wedge t_1, t_2 \in \mathcal{A}\}$$

```
1 x ← [-∞, +∞]
2 y ← input
while 3(x > 0) do
    4 x ← x - y
od5
```



Piecewise-Defined Ranking Functions Abstract Domain

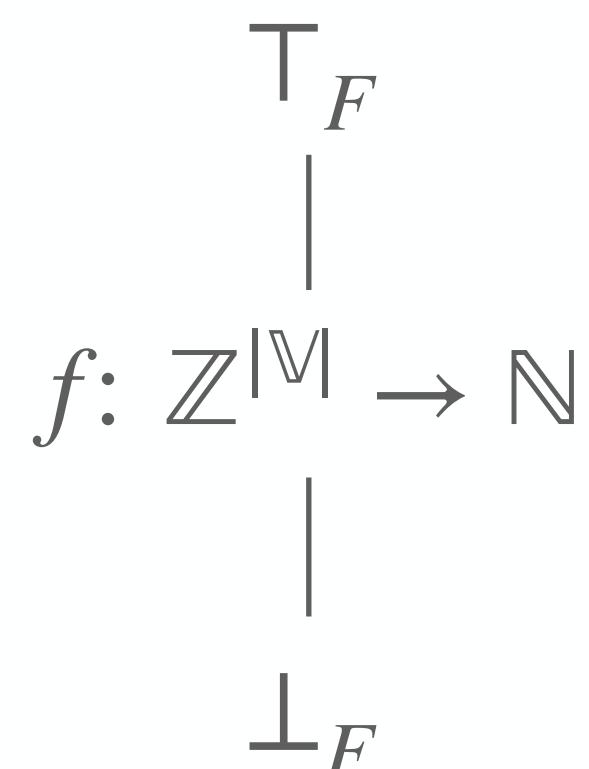
Functions Auxiliary Abstract Domain

- computational order $\sqsubseteq_F[D]$, where $D \in \mathcal{D}$:

- between defined leaf nodes:

$$f_1 \sqsubseteq_F [D] f_2 \stackrel{\text{def}}{=} \forall \rho \in \gamma_D(D) : f_1(\dots, \rho(X_i), \dots) \leq f_2(\dots, \rho(X_i), \dots)$$

- otherwise (i.e., when one or both leaf nodes are undefined):



Piecewise-Defined Ranking Functions Abstract Domain

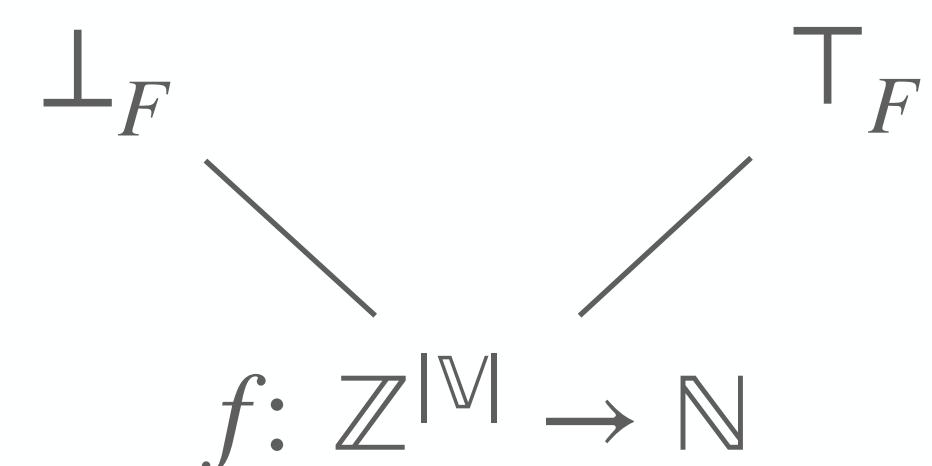
Functions Auxiliary Abstract Domain

- **approximation order** $\leqslant_F [D]$, where $D \in \mathcal{D}$:

- between defined leaf nodes:

$$f_1 \leqslant_F [D] f_2 \stackrel{\text{def}}{=} \forall \rho \in \gamma_D(D) : f_1(\dots, \rho(X_i), \dots) \leq f_2(\dots, \rho(X_i), \dots)$$

- otherwise (i.e., when one or both leaf nodes are undefined):



Piecewise-Defined Ranking Functions Abstract Domain

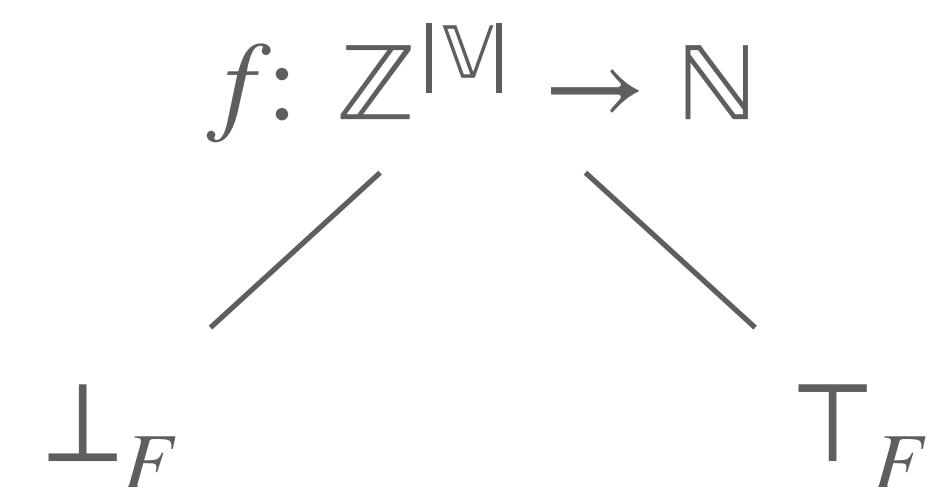
Functions Auxiliary Abstract Domain

- **resilience order** $\leqslant_F [D]$, where $D \in \mathcal{D}$:

- between defined leaf nodes:

$$f_1 \leqslant_F [D] f_2 \stackrel{\text{def}}{=} \forall \rho \in \gamma_D(D) : f_1(\dots, \rho(X_i), \dots) \geq f_2(\dots, \rho(X_i), \dots)$$

- otherwise (i.e., when one or both leaf nodes are undefined):



Piecewise-Defined Ranking Functions Abstract Domain

Abstract Domain Operators

- They manipulate elements in $\mathcal{A}_{\text{NIL}} \stackrel{\text{def}}{=} \{\text{NIL}\} \cup \mathcal{A}$
- The **binary operators** rely on a tree unification algorithm
 - approximation order \leq_A , computational order \sqsubseteq_A , and **resilience order** \leq_A
 - approximation join γ_A , computational join \sqcup_A , and **resilience join** \vee_A
 - meet λ_A
 - widening ∇_A
- The **unary operators** rely on a tree pruning algorithm
 - assignment $\overleftarrow{\text{ASSIGN}}_A[X \leftarrow e]$
 - test $\text{FILTER}_A[e]$

Piecewise-Defined Ranking Functions Abstract Domain

Resilience Order

1. Perform **tree unification**
2. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C
3. Compare the leaf nodes using the **resilience order** $\leq_F[\alpha_C(C)]$

Piecewise-Defined Ranking Functions Abstract Domain

Resilience Join

1. Perform **tree unification**
2. Recursively descend the trees while *accumulating the linear constraints encountered along the paths* into a set of constraints C
3. $\text{NIL} \vee_A t \stackrel{\text{def}}{=} t$
 $t \vee_A \text{NIL} \stackrel{\text{def}}{=} t$
4. Join the leaf nodes using the **resilience join** $\underline{\vee}_F [\alpha_C(C)]$

Piecewise-Defined Ranking Functions Abstract Domain

Resilience Join (continue)

- **resilience join** $\vee_F [D]$, where $D \in \mathcal{D}$:

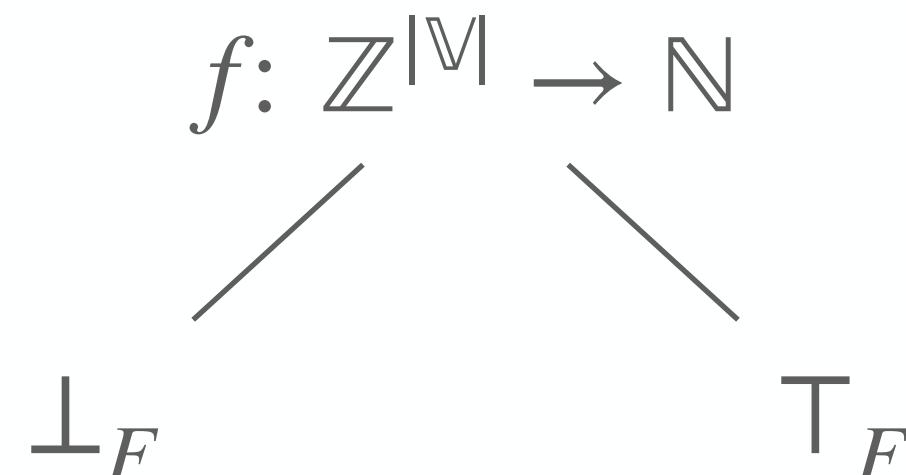
- between defined leaf nodes:

$$f_1 \vee_F [D] f_2 \stackrel{\text{def}}{=} \begin{cases} f & f \in \mathcal{F} \setminus \{ \perp_F, \top_F \} \\ \top_F & \text{otherwise} \end{cases}$$

where $f \stackrel{\text{def}}{=} \lambda \rho \in \gamma_D(D) : \min(f_1(\dots, \rho(X_i), \dots), f_2(\dots, \rho(X_i), \dots))$

- otherwise (i.e., when one or both leaf nodes are undefined):

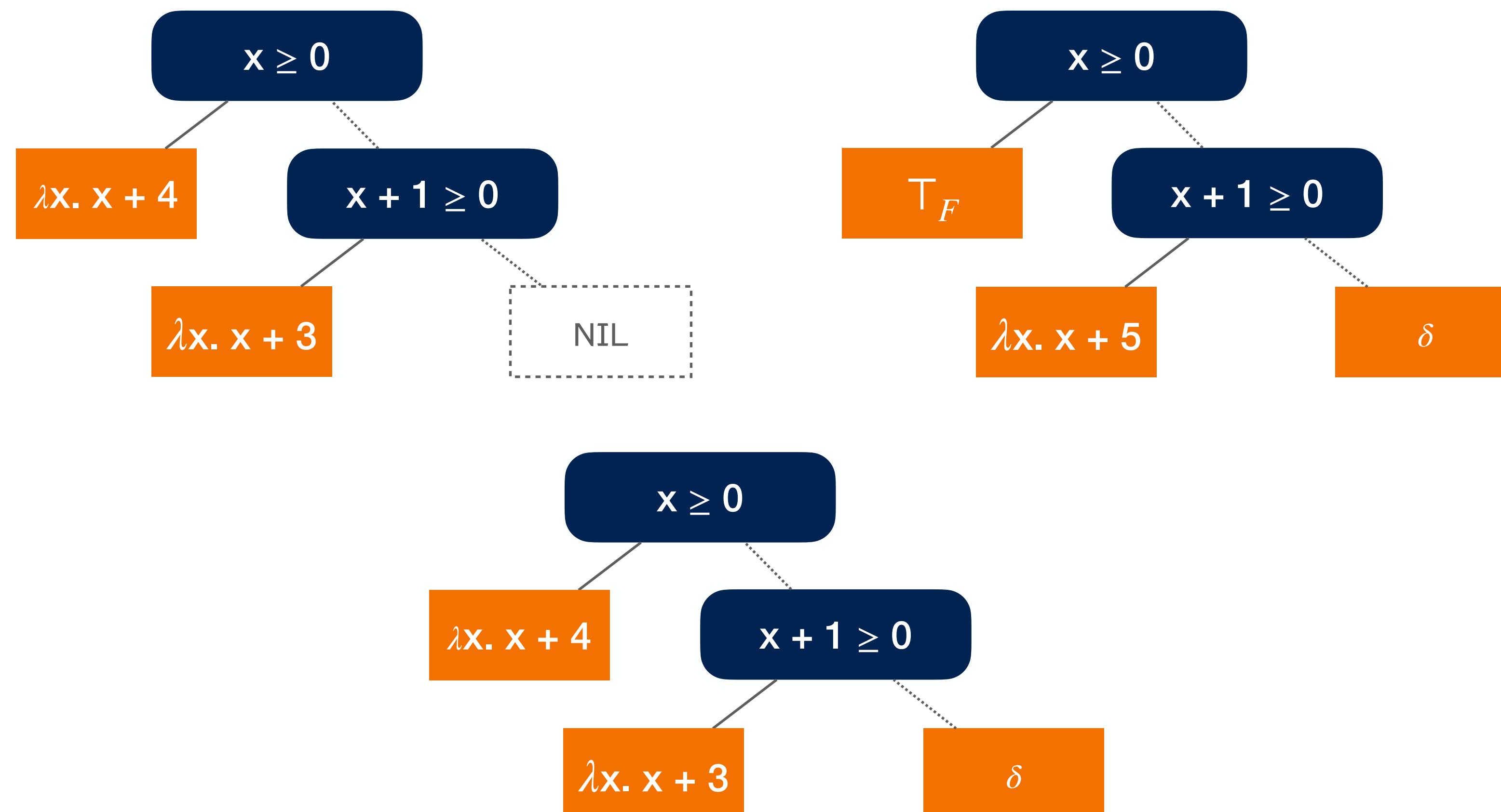
$$\begin{array}{ll} \perp_F \vee_F [D] f \stackrel{\text{def}}{=} f & f \in \mathcal{F} \setminus \{ \top_F \} \\ f \vee_F [D] \perp_F \stackrel{\text{def}}{=} f & f \in \mathcal{F} \setminus \{ \top_F \} \\ \top_F \vee_F [D] f \stackrel{\text{def}}{=} f & f \in \mathcal{F} \setminus \{ \perp_F \} \\ f \vee_F [D] \top_F \stackrel{\text{def}}{=} f & f \in \mathcal{F} \setminus \{ \perp_F \} \end{array}$$



Piecewise-Defined Ranking Functions Abstract Domain

Resilience Join

Example



Piecewise-Defined Ranking Functions Abstract Domain

Assignments

- Base case (f)

$\overleftarrow{\text{ASSIGN}}_A[X \leftarrow \text{input}]$

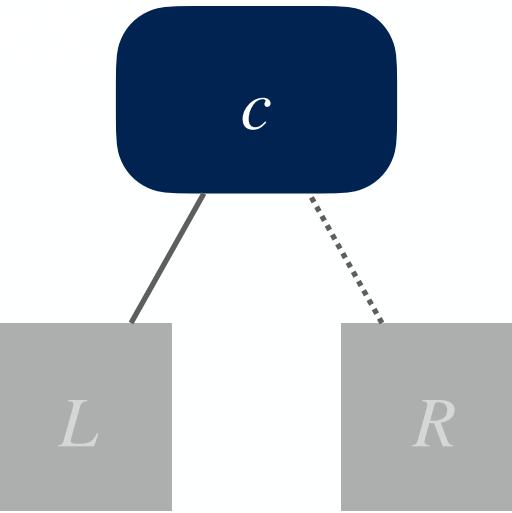
Apply $\overleftarrow{\text{ASSIGN}}_F[X \leftarrow \text{input}][\alpha_C(C)]$ on the defined leaf nodes

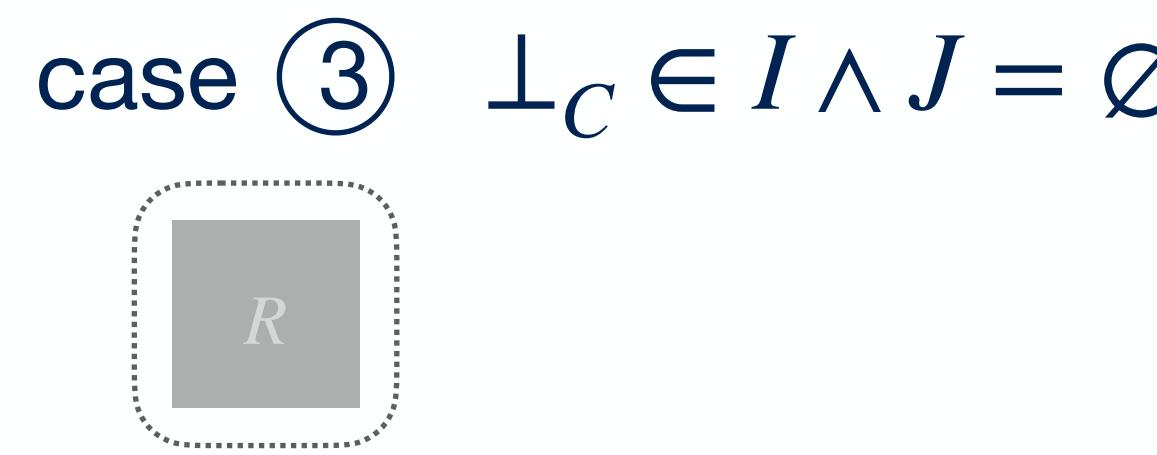
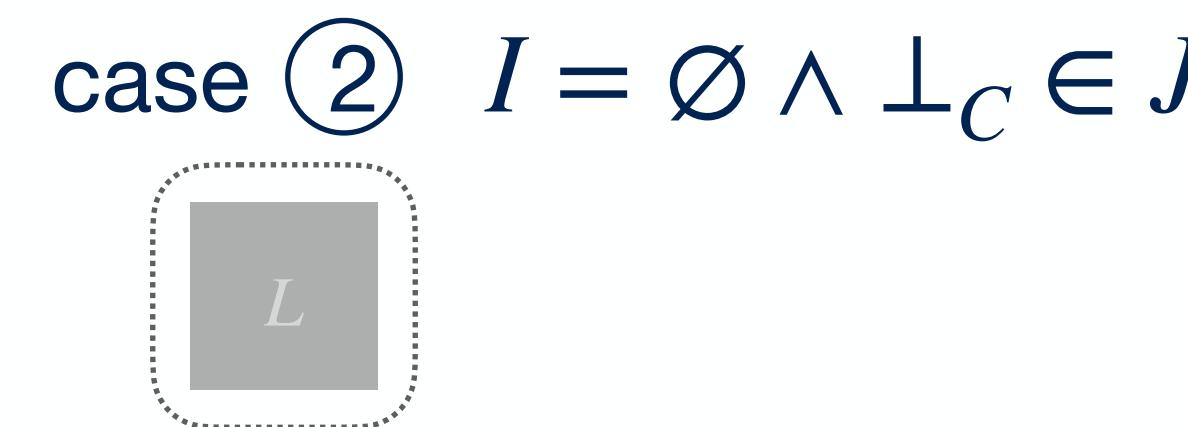
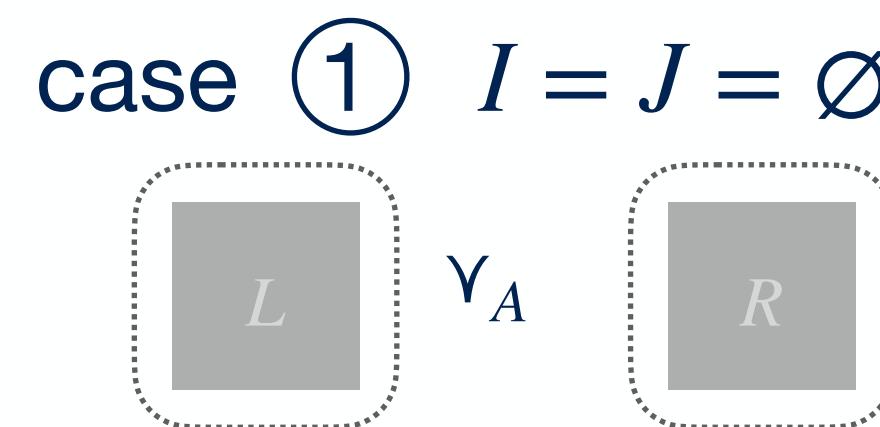
$$\overleftarrow{\text{ASSIGN}}_F[X \leftarrow \text{input}][D](f) \stackrel{\text{def}}{=} \begin{cases} \bar{f} & \bar{f} \in \mathcal{F} \setminus \{ \perp_F, \top_F \} \\ \top_F & \text{otherwise} \end{cases} \quad f \in \mathcal{F} \setminus \{ \perp_F, \top_F \}$$

where $\bar{f}(\dots, X_i, X, \dots) \stackrel{\text{def}}{=} \max\{f(\dots, \rho(X_i), v, \dots) + 1 \mid \rho \in \gamma_D(R) \wedge v \in E[e]\rho\}$
and $R \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_D[X \leftarrow [-\infty, +\infty]]D$

Piecewise-Defined Ranking Functions Abstract Domain

Assignments

-  Convert $\overleftarrow{\text{ASSIGN}}_D[X \leftarrow [-\infty, +\infty]](\alpha_C(\{c\}) \text{ and } \overleftarrow{\text{ASSIGN}}_D[X \leftarrow [-\infty, +\infty]](\alpha_C(\{\neg c\}))$ into sets I and J of linear constraints in canonical form



- case ④
1. perform **tree pruning** on 
 2. join the results with the **approximation join** \vee_A

$\overleftarrow{\text{ASSIGN}}_A[X \leftarrow \text{input}]$

Piecewise-Defined Ranking Functions Abstract Domain

Assignments

- Base case (f)

$\overleftarrow{\text{ASSIGN}}_A[X \leftarrow e]$

Apply $\overleftarrow{\text{ASSIGN}}_F[X \leftarrow e][\alpha_C(C)]$ on the defined leaf nodes
 the value of e (and thus of X) depends from the input values read by the program

$$\overleftarrow{\text{ASSIGN}}_F[X \leftarrow e][D](f) \stackrel{\text{def}}{=} \begin{cases} \bar{f} & \text{TAIN}[e] \wedge \bar{f} \in \mathcal{F} \setminus \{ \perp_F, \top_F \} \\ f & \neg \text{TAIN}[e] \wedge f \in \mathcal{F} \setminus \{ \perp_F, \top_F \} \\ \top_F & \text{otherwise} \end{cases} \quad f \in \mathcal{F} \setminus \{ \perp_F, \top_F \}$$

where $\bar{f}(\dots, X_i, X, \dots) \stackrel{\text{def}}{=} \max\{f(\dots, \rho(X_i), v, \dots) + 1 \mid \rho \in \gamma_D(R) \wedge v \in E[e]\rho\}$

$\underline{f}(\dots, X_i, X, \dots) \stackrel{\text{def}}{=} \min\{f(\dots, \rho(X_i), v, \dots) + 1 \mid \rho \in \gamma_D(R) \wedge v \in E[e]\rho\}$

and $R \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_D[X \leftarrow e]D$

Simple Taint Analysis

$\text{Taint}[\text{stat}] : \mathcal{P}(\mathbb{X}) \rightarrow \mathcal{P}(\mathbb{X})$

$$\text{Taint}[\ell X \leftarrow \text{input}]T \stackrel{\text{def}}{=} T \cup \{X\}$$

$$\text{Taint}[\ell X \leftarrow e]T \stackrel{\text{def}}{=} \begin{cases} T \cup \{X\} & \text{Taint}[e]T \\ T \setminus \{X\} & \text{otherwise assigned variables in } s \end{cases}$$

$$\text{Taint}[\text{if } \ell e \bowtie 0 \text{ then } s]T \stackrel{\text{def}}{=} \begin{cases} \text{Taint}[s]T \cup \text{LHS}(s) \cup T & \text{Taint}[e]T \\ \text{Taint}[s]T \cup T & \text{otherwise} \end{cases}$$

$$\text{Taint}[\text{while } \ell e \bowtie 0 \text{ do } s \text{ done}]T \stackrel{\text{def}}{=} \text{lfp}_T^{\subseteq} \left(\lambda X. \begin{cases} \text{Taint}[s]X \cup \text{LHS}(s) \cup X & \text{Taint}[e]X \\ \text{Taint}[s]X \cup X & \text{otherwise} \end{cases} \right)$$

$$\text{Taint}[s_1; s_2]T \stackrel{\text{def}}{=} \text{Taint}[s_2](\text{Taint}[s_1]T)$$

$$\text{Taint}[e]T \stackrel{\text{iff}}{\leftrightarrow} \text{VARS}(e) \cap T \neq \emptyset$$

variables in e

Language syntax

ℓstat	$::= \ell X \leftarrow \text{exp}$ $\quad \quad \ell \text{if } \text{exp} \bowtie 0 \text{ then } \ell \text{stat}$ $\quad \quad \ell \text{while } \ell \text{exp} \bowtie 0 \text{ do } \ell \text{stat}$ $\quad \quad \ell \text{stat}; \ell \text{stat}$ $\quad \quad \ell \text{done}$	(assignment) (conditional) (loop) (sequence)
exp	$::= X$ $\quad \quad -\text{exp}$ $\quad \quad \text{exp} \diamond \text{exp}$ $\quad \quad c$ $\quad \quad [c, c']$	(variable) (negation) (binary operation) (constant $c \in \mathbb{Z}$) (random input, $c, c' \in \mathbb{Z} \cup \{\pm\infty\}$)

Simple structured, numeric language

- $X \in \mathbb{V}$, where \mathbb{V} is a finite set of program variables
- $\ell \in \mathcal{L}$, where \mathcal{L} is a finite set of control points
- numeric expressions: $\bowtie \in \{=, \leq, \dots\}$, $\diamond \in \{+, -, \times, /\}$
- random inputs: $X \leftarrow [c, c']$
model environment, parametric programs, unknown functions, ...

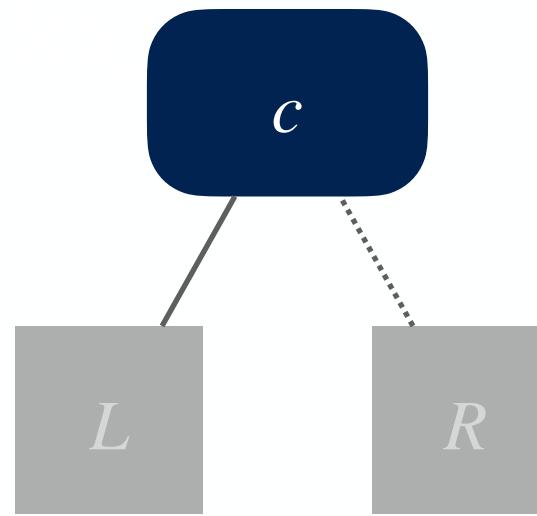
Course 2 Program Semantics and Properties Antoine Mine

p. 3 / 98

Piecewise-Defined Ranking Functions Abstract Domain

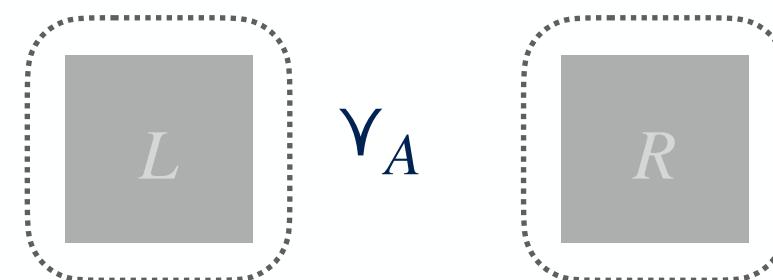
Assignments

-

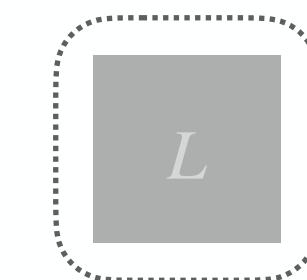


Convert $\overleftarrow{\text{ASSIGN}}_D[X \leftarrow e](\alpha_C(\{c\}))$ and $\overleftarrow{\text{ASSIGN}}_D[X \leftarrow e](\alpha_C(\{\neg c\}))$ into sets I and J of linear constraints in canonical form

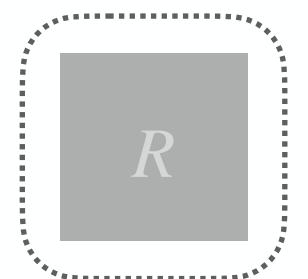
case ① $I = J = \emptyset$



case ② $I = \emptyset \wedge \perp_C \in J$



case ③ $\perp_C \in I \wedge J = \emptyset$



case ④

1. perform **tree pruning** on and
2. join the results with Y_A (if $\text{TAINT}[e]$) or \underline{Y}_A (if $\neg \text{TAINT}[e]$)

$\overleftarrow{\text{ASSIGN}}_A[X \leftarrow e]$

Abstract Termination Resilience Semantics

Piecewise-Defined Ranking Functions Abstract Domain

For each program instruction stat, we define a transformer $\mathcal{R}_{TR}^\# \llbracket \text{stat} \rrbracket : \mathcal{A} \rightarrow \mathcal{A}$:

- $\mathcal{R}_{TR}^\# \llbracket \ell X \leftarrow \text{input} \rrbracket t \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_A \llbracket X \leftarrow \text{input} \rrbracket t$
- $\mathcal{R}_{TR}^\# \llbracket \ell X \leftarrow e \rrbracket t \stackrel{\text{def}}{=} \overleftarrow{\text{ASSIGN}}_A \llbracket X \leftarrow e \rrbracket t$
- $\mathcal{R}_{TR}^\# \llbracket \text{if } \ell e \bowtie 0 \text{ then } s \rrbracket t \stackrel{\text{def}}{=} \begin{cases} \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket (\mathcal{R}_{TR}^\# \llbracket s \rrbracket t) \vee_T \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket t & \text{TAIN}_A \llbracket e \rrbracket \\ \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket (\mathcal{R}_{TR}^\# \llbracket s \rrbracket t) \vee_T \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket t & \text{otherwise} \end{cases}$
- $\mathcal{R}_{TR}^\# \llbracket \text{while } \ell e \bowtie 0 \text{ do } s \text{ done} \rrbracket t \stackrel{\text{def}}{=} \text{lfp}^\# \bar{F}_{TR}^\#$
where $\bar{F}_{TR}^\#(x) \stackrel{\text{def}}{=} \begin{cases} \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket (\mathcal{R}_{TR}^\# \llbracket s \rrbracket x) \vee_T \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket (t) & \text{TAIN}_A \llbracket e \rrbracket \\ \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket (\mathcal{R}_{TR}^\# \llbracket s \rrbracket x) \vee_T \text{FILTER}_A \llbracket e \bowtie 0 \rrbracket (t) & \text{otherwise} \end{cases}$
- $\mathcal{R}_{TR}^\# \llbracket s_1 ; s_2 \rrbracket t \stackrel{\text{def}}{=} \mathcal{R}_{TR}^\# \llbracket s_1 \rrbracket (\mathcal{R}_{TR}^\# \llbracket s_2 \rrbracket t)$

Abstract Termination Resilience Semantics

Piecewise-Defined Ranking Functions Abstract Domain

Definition

The **abstract termination resilience semantics** $\mathcal{R}_{TR}^{\#}[\![\text{stat}^{\ell}]\!] \in \mathcal{A}$ of a program stat^{ℓ} is:

$$\mathcal{R}_{TR}^{\#}[\![\text{stat}^{\ell}]\!] \stackrel{\text{def}}{=} \mathcal{R}_{TR}^{\#}[\![\text{stat}]\!](\text{LEAF}: \lambda X_1, \dots, X_k. 0)$$

where $\mathcal{R}_{TR}^{\#}[\![\text{stat}]\!]: \mathcal{A} \rightarrow \mathcal{A}$ is the abstract termination resilience semantics of each instruction stat

Theorem (Soundness)

$$\mathcal{R}_{TR}[\![\text{stat}^{\ell}]\!] \leq \gamma_A(\mathcal{R}_{TR}^{\#}[\![\text{stat}^{\ell}]\!])$$

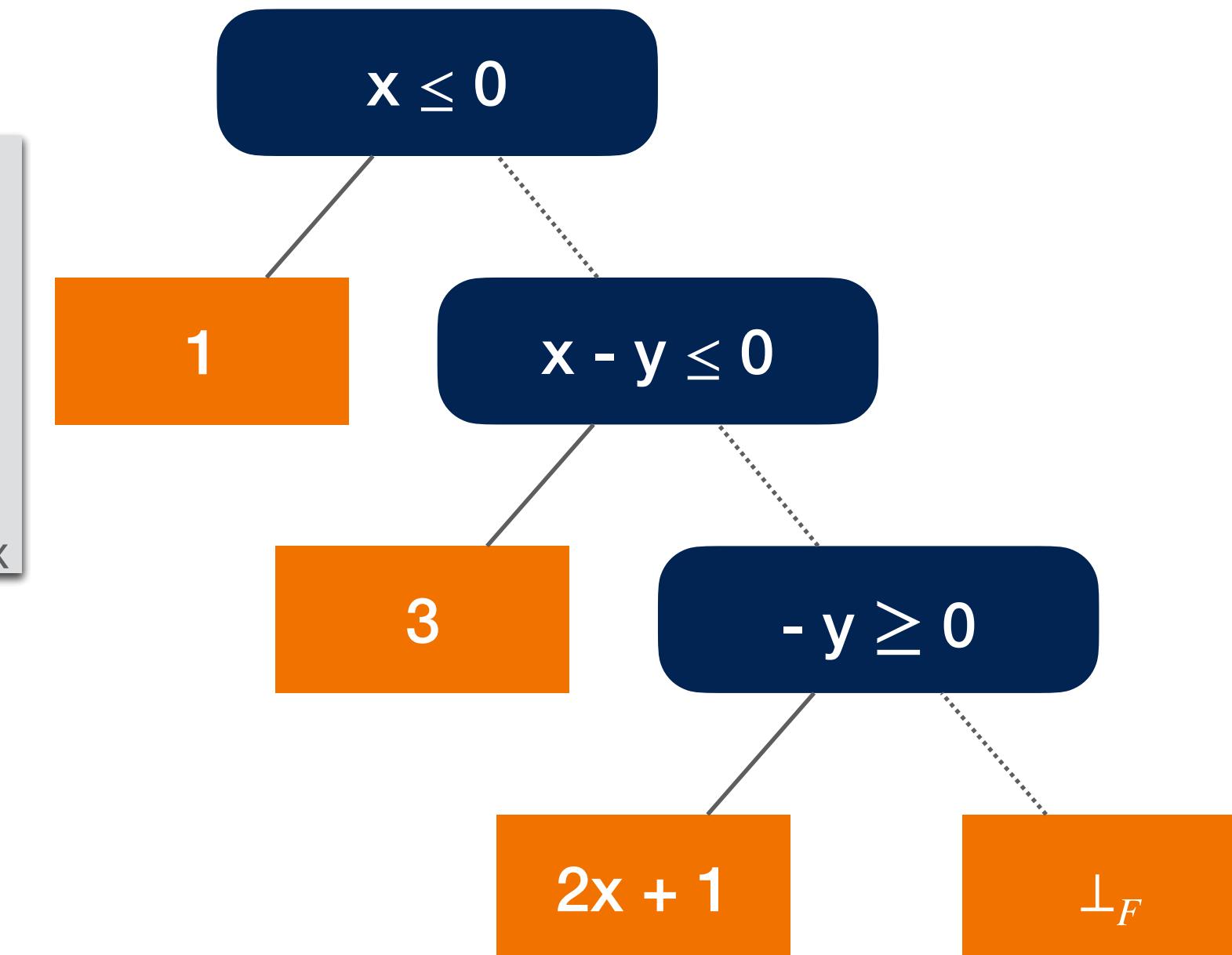
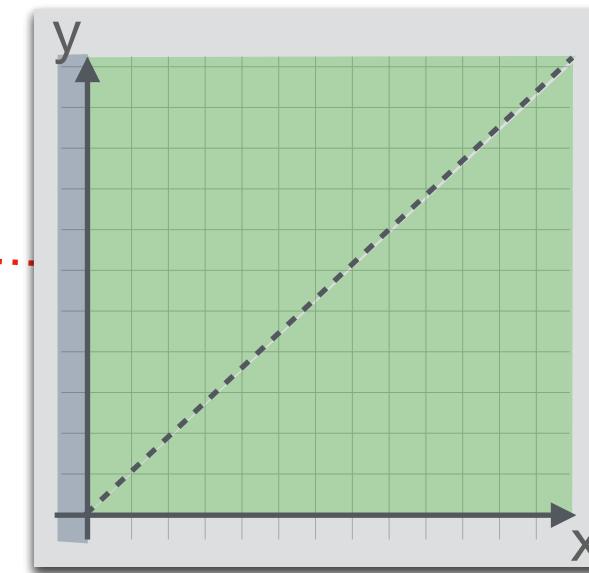
Corollary (Soundness)

A program stat^{ℓ} **satisfies termination resilience** for traces starting from a set of initial states \mathcal{I} , i.e., $\mathcal{M}_{\infty}[\![\text{stat}^{\ell}]\!](\mathcal{I}) \in \text{TerminationResilience}$, if $\mathcal{I} \subseteq \text{dom}(\gamma_A(\mathcal{R}_{TR}^{\#}[\![\text{stat}^{\ell}]\!]))$

Abstract Termination Resilience Semantics

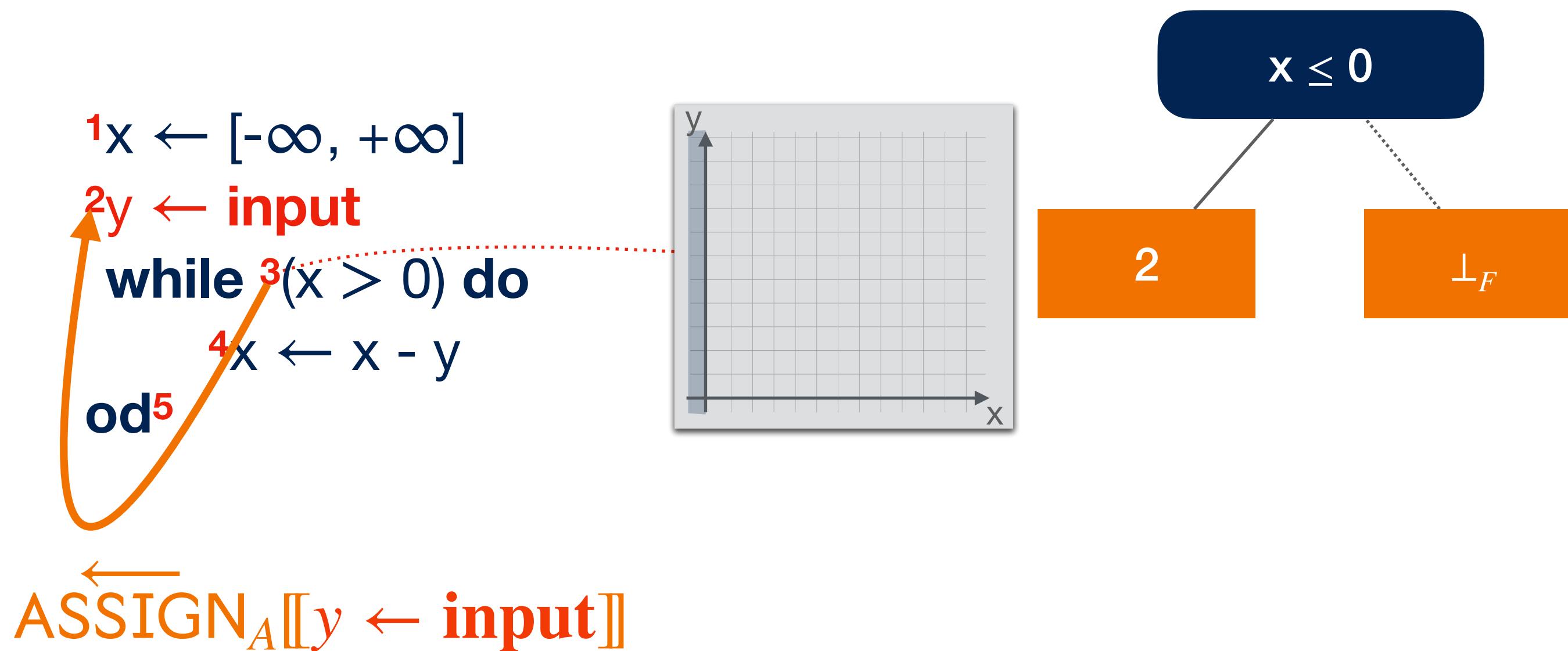
Example

```
1x ← [-∞, +∞]  
2y ← input  
while 3(x > 0) do  
  4x ← x - y  
od5
```



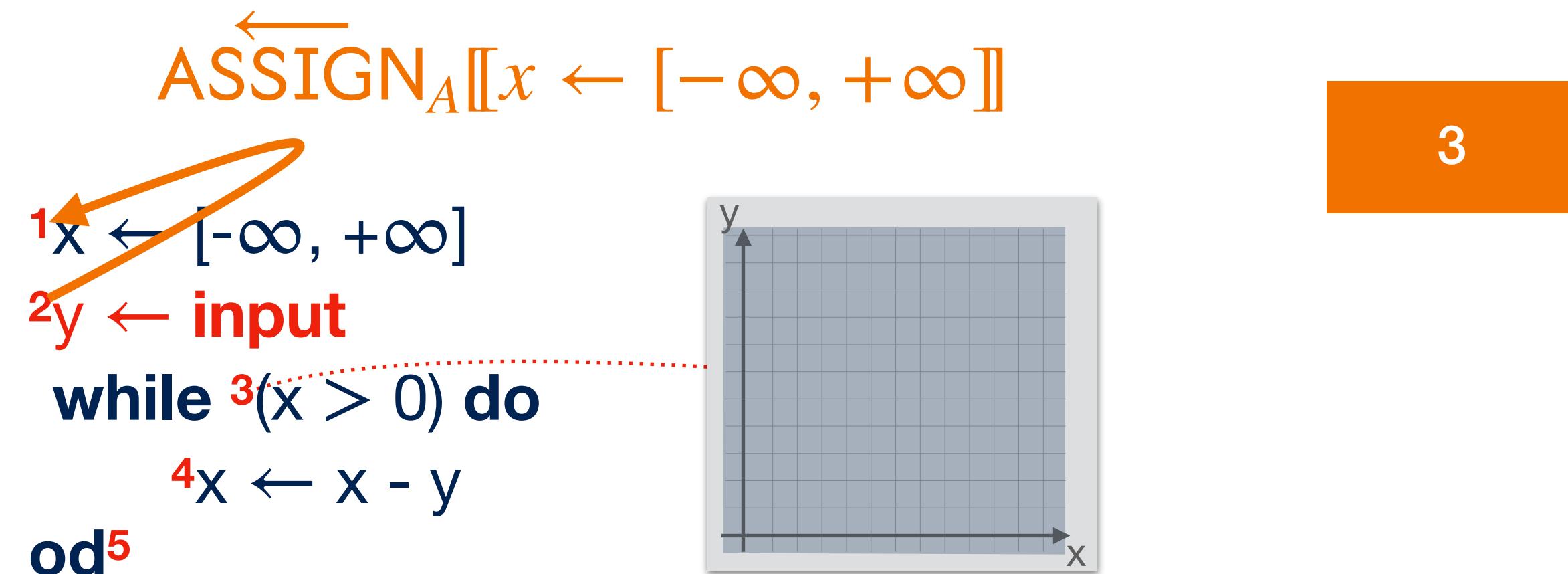
Abstract Termination Resilience Semantics

Example

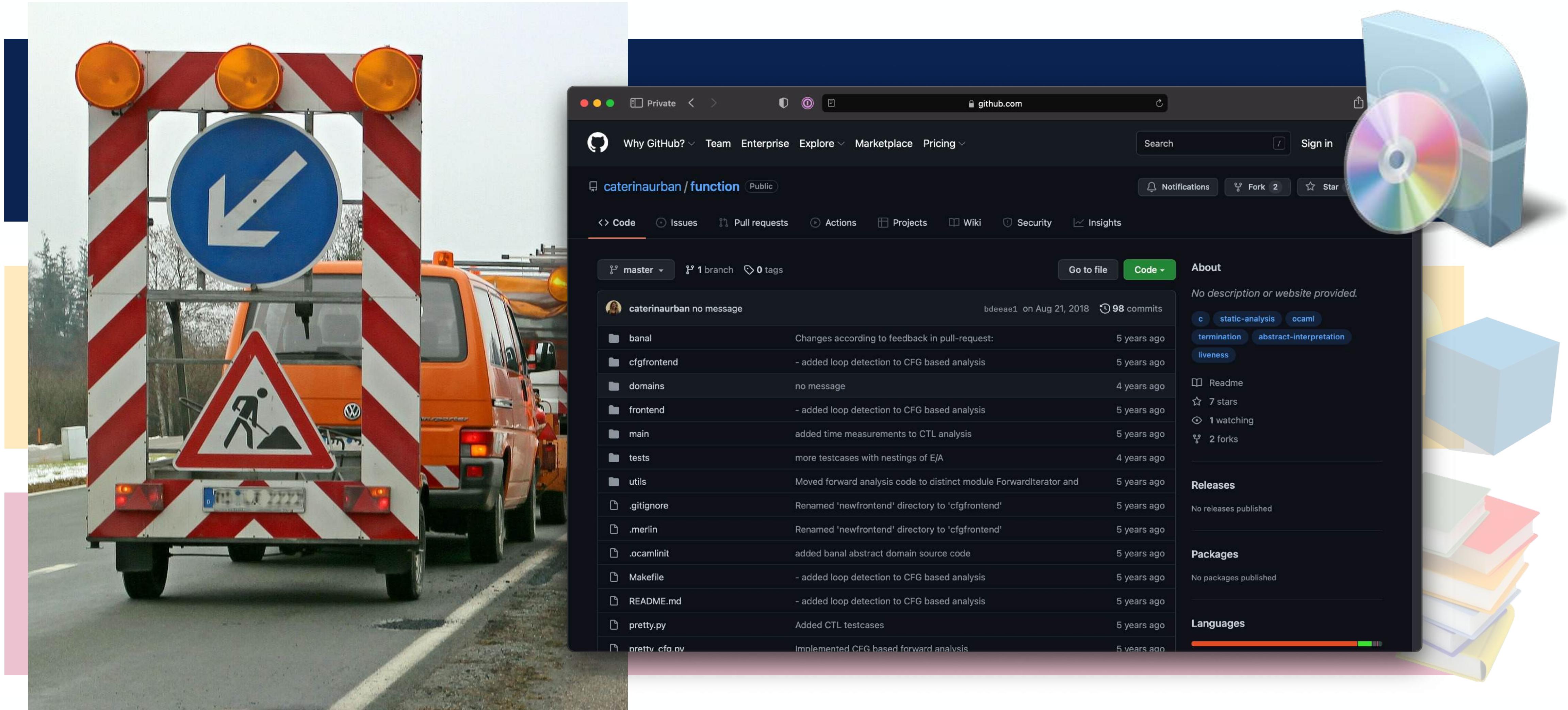


Abstract Termination Resilience Semantics

Example



Implementation



Bibliography

[Moussaoui24] **Naïm Moussaoui Remil and Caterina Urban.** Termination Resilience Static Analysis.
Draft, 2024.

if interested, ask me for a **draft**